ARMY RESEARCH LABORATORY

Process for Automated, Safe MBE Start and Flux Calibration

Stefan Svensson

ARL-TR-2361 February 2001

Army Research Laboratory

Adelphi, MD 20783-1197

ARL-TR-2361 February 2001

Process for Automated, Safe MBE Start and Flux Calibration

Stefan Svensson

Sensors and Electron Devices Directorate

Abstract

A command procedure has been developed for the U.S. Army Research Laboratory (ARL) molecular beam epitaxy (MBE) computer control system that allows a user to set up the system for an automated, unattended start each morning. The automated sequence consists of-

1. A system safety check to determine if cell ramping should be allowed.
2. A cell temperature ramp to an outgassing temperature.
3. An outgassing of cells.
4. A ramp-down of cells to nominal operating temperatures.
5. An automated setup through an iterative process of flux measurements and changes of temperatures until desired targets are reached.

This command procedure allows a daily, safe start-up of the MBE system and generates identical flux settings that improve the crystal growth reproducibility. Typically, one can save two hours or more of a work day by using this automated procedure.

Contents

Figure

1. Background

The U.S. Army Research Laboratory (ARL) molecular beam epitaxy (MBE) system is controlled by a PC-based system called "Molly," which is supplied by EPI Technologies, Inc. Molly provides a script language that can be used to create command procedures that execute customized sequences of actions on the MBE system. Possible actions are reading and setting cell temperatures, opening and closing shutters, reading pressure gauges, and turning the substrate holder.

The PC-based control system replaced an older PDP-11 system. The shortcomings of this latter system were clarified after a malfunction during which the MBE machine had been programmed to start a cell up-ramp when the liquid nitrogen had been inadvertently turned off. This caused damage to the growth system, requiring venting and replacement of some cells. Although the PDP-11-based system could read pressures, it did not allow decisions to be programmed in to make actions conditional on any system status parameter. Molly provided a solution to this problem but required custom written code. This report describes the result of that effort.

2. Flow Chart

The logic behind the developed script is that the MBE system is idling over night, with the evaporation cells at a low temperature at which the evaporation rate is negligible. After the system vacuum is checked to ensure a safe up-ramp, the cells are slowly brought up to an outgassing temperature above the estimated set points for growth. The up-ramp is typically 0.5 hr, to allow the cells to thermalize to avoid stresses. At the peak temperature, the shutters are opened for about 10 min. to allow material that may have condensed, at or near each cell at the idling temperatures, to be evaporated so as to provide cleaner molecular beams during growth. After outgassing, the shutters are closed and the cell temperatures are lowered to the previous day's set points.

Because material is consumed during growth and the temperature sensor in each cell does not perfectly represent the melt temperature from day to day, the previous day's set points typically do not exactly reproduce the previous day's fluxes. The set points must therefore be changed based on the difference between measured fluxes and the target. All cells obey a linear relationship between the logarithm of the flux and the inverse of the absolute temperature of the cell. This relationship is used to calculate the needed temperature change based on the measured flux difference. After a new temperature is set, the computer is programmed to wait a predetermined time to let the cell reach equilibrium before a new measurement is taken. Some hysteresis is typically experienced in this process that requires up to about six repetitions before acceptable accuracy is reached. The accuracy (| (target-measured)/target |) is a variable that is typically set to 0.0025—a level of precision for which a human operator seldom can muster the patience.

The actual process of measuring the fluxes has been designed to avoid the flux transients typically seen when shutters are opened. These transients are caused by the fact that with the shutter in closed position, heat from the melt surface is radiated back from the shutter to the melt. When the shutter is abruptly opened, the steady state is interrupted and heat radiation is lost from the melt at a higher rate, resulting in a drop in the flux. After some time, the thermocouple at the bottom of the crucible experiences a drop in the melt temperate, prompting the controller to increase the power to the cell until the temperature set point is restored. After this control sequence has reached a new steady state, the flux is stable. The measurement of the flux must consequently be done at this point or later and not during the transient.

The ionization gauge used for flux measurements is turned away from the cells when it is not used for measurements to increase its lifetime. A flux measurement sequence thus consists of (1) opening the shutter for a predetermined time (usually 10 min.), (2) turning the gauge toward the cell, (3) averaging of 10 flux readings to determine the flux plus the background pres-

sure, (4) closing the shutter, (5) waiting for the gauge reading to stabilize, (6) averaging of 10 flux readings to determine the chamber background pressure, (7) subtracting the second reading from the first to obtain the net flux, and (8) turning away the gauge from the cells again. If the measured and target fluxes deviate more than the preset accuracy, a new temperature is calculated and set. The system then waits for the cell to stabilize at the new temperature.

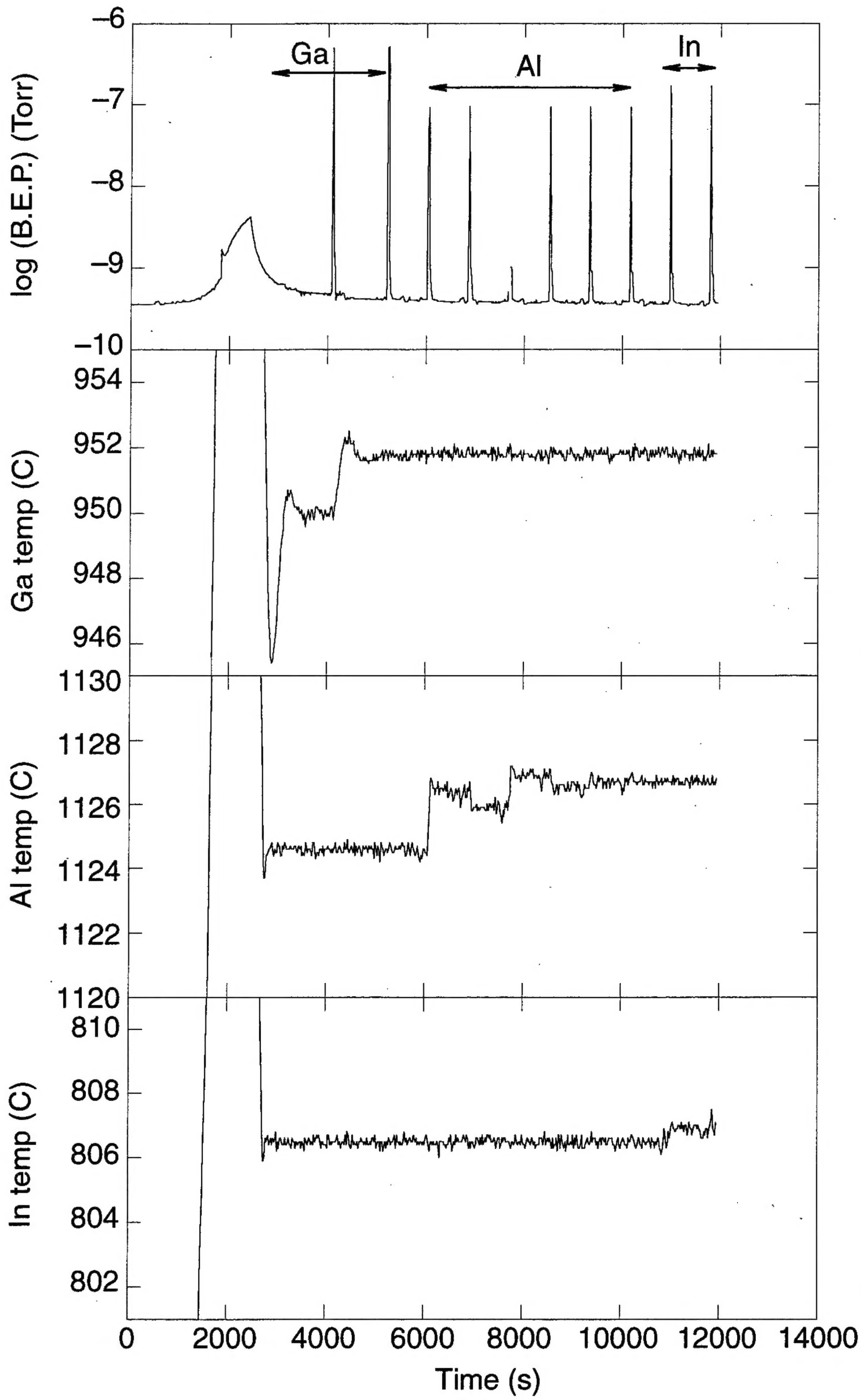
During the calibration sequence, the pressures and temperatures are logged in a standard file. The data in this file can be extracted and plotted as in figure 1 on page 4. A good practice is to save the log file with the day's date for future reference.

For the script file to properly execute, certain information must be provided. The file (Calib.cmd) can be opened with any text editor. I have used WinEdit, which is a shareware program editor. The script file has been written with enough comments next to the variable declarations to indicate what needs to be entered. Typical inputs are—

- Which cells are to be ramped.
- To what temperature the cells are to be ramped.
- If the cells are or are not to be included in the flux calibration sequence.
- What the flux targets are.
- The start time of the ramps.

Optionally, other parameters can be changed, although the default values normally provide good performance. These parameters include the calibration accuracy, the slope and intercept of the flux versus temperature lines, the outgassing temperature expressed as a percentage above the growth set point, and the chamber pressure that cannot be exceeded if ramping is to be started. The Appendix contains the script listing.

Figure 1. Logged flux and temperature data as a function of time. Time up to 2500 s is spent on up-ramp and outgassing. At 2500 s, previous day's set points have been reached and flux measurement and correction sequence starts. (First Ga flux reading ~3000 s is missing because of sampling rate in file being too low.)



3. Summary and Conclusion

Control code has been developed to allow unattended start-up of an MBE system. The code has been used and tested numerous times and delivered very accurate growth parameters and subsequent crystal layers with extremely small thickness and composition errors. In at least one case, the cell up-ramp was not started because of an excessively high-growth chamber pressure, thus preventing potential harm to the system.

Appendix. Script Listings

The following is a script listing of commands that execute a customized sequence of actions on the MBE system.

```
/*
/************************************************************************************/
/*                                                                                  */
/*   This command file performs a growth chamber pressure check, a cell up-ramp,    */
/*   a cell degas, and flux calibration starting at a predetermined time.            */
/*                                                                                  */
/*                                   Stefan Svensson, ARL  May 7, 1997              */
/************************************************************************************/
/*
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
#include <cells.h>
#include <shutters.h>
#include <mbe.h>
/**/
/*=================================================================================*/
/*==========         Declarations and initial values          ==========*/
/*=================================================================================*/
/**/

/*    START BY FLAGGING THE CELLS TO BE RAMPED                                              */

int rampGa  = 1;                        /* Ramp flag for Ga  (1=yes 0 = no)         */
int rampAl3 = 0;                        /* Ramp flag for Al3 (1=yes 0 = no)         */
int rampAl4 = 1;                        /* Ramp flag for Al4 (1=yes 0 = no)         */
int rampIn  = 1;                        /* Ramp flag for In  (1=yes 0 = no)         */
int rampSi  = 1;                        /* Ramp flag for Si  (1=yes 0 = no)         */
int rampBe  = 0;                        /* Ramp flag for Be  (1=yes 0 = no)         */
int rampSb  = 0;                        /* Ramp flag for Sb  (1=yes 0 = no)         */

/*    THEN ENTER THEIR SET POINTS (they will be outgassed at a 2.5% higher temp)         */

double setpGa  =  956.3;                /* Target temp for Ga       ADJUST          */
double setpAl3 = 1135.2;                /* Target temp for Al3      BASED           */
double setpAl4 = 1111.1;                /* Target temp for Al4      ON              */
double setpIn  =  790.7;                /* Target temp for In       PREVIOUS        */
double setpSi  = 1327.4;                /* Target temp for Si       CAL             */
double setpBe  =  922.5;                /* Target temp for Be                       */
double setpSb  =  400.0;                /* Target temp for Sb                       */

/*     THEN FLAG THE CELLS WHICH WILL BE FLUX CALIBRATED                                  */

int calGa  = 1;                         /* Calib flag for Ga  (1=yes 0 = no)        */
int calAl3 = 0;                         /* Calib flag for Al3 (1=yes 0 = no)        */
int calAl4 = 1;                         /* Calib flag for Al4 (1=yes 0 = no)        */
int calIn  = 1;                         /* Calib flag for In  (1=yes 0 = no)        */
int calSb  = 0;                         /* Calib flag for Sb  (1=yes 0 = no)        */

/*     THEN ENTER THE FLUX TARGET VALUES                                                  */

double targetGa  = 5.25E-7;             /* Target flux for Ga                      */
double targetAl3 = 1.062E-7;            /* Target flux for Al3                      */
double targetAl4 = 0.844E-7;              /* Target flux for Al4                    */
double targetIn  = 1.63E-7;              /* Target flux for In                      */
double targetSb  = 1.0E-7;               /* Target flux for Sb                      */

/*   AND THE PRECISION OF THE CALIBRATION (cal ends when abs( (flux-target)/target ) < prec */

double precGa  = 0.0025;                 /* Target presision for Ga                 */
double precAl3 = 0.0025;                 /* Target presision for Al3                */
double precAl4 = 0.0025;                 /* Target presision for Al4                */
--double precIn  = 0.0025;                 /* Target presision for In                 */
double precSb  = 0.0025;                 /* Target presision for Sb                 */
```

```
/*     FINALLY, DECIDE WHEN TO START THE EXECUTION                              */

/*  Use the formula:                                                            */
/*  Finish time = Start time + 45 + 30*number of cells to cal (min)             */
/*     (The actual time depends on accuracy of initial set point and desired precision    */

int hour   = 8;                         /* Start hour use 24 hr clock (0-23 valid) */
int minute = 32;                        /* Start minute                            */

/*     DON'T FORGET TO PUT SHUTTERS AND CAR IN REMOTE                           */

/*     END OF STANDARD ENTRIES                                                  */

double backtest   = 2.E-9;             /* Background pressure for up-ramp test     */
double ramptime   = 30.0;               /* Length of cell ramp  (min)               */
double transient  = 150.0;              /* Wait time to avoid shutter transient (sec)*/
double outgastime = 10.0;               /* Outgas time after up-ramp  (min)         */
double outgas     = 2.5;                /* Percent temp increase to outgas at       */

double slopeGa  = -12670.0;             /* Flux slope for Ga        ADJUST        (prev -11630) */
double slopeAl3 = -10000.0;             /* Flux slope for Al3       ONLY IF       */
double slopeAl4 = -12670.0;             /* Flux slope for Al4       CAL DOES      (prev -13860) */
double slopeIn  = -15707.0;             /* Flux slope for In        NOT CONVERGE  (prev -12080) */
double slopeSb  = -10000.0;             /* Flux slope for Sb        FAST ENOUGH   */

double stabGa  = 900.;                  /* Stabilization time for Ga               */
double stabAl3 = 600.;                  /* Stabilization time for Al3              */
double stabAl4 = 600.;                  /* Stabilization time for Al4              */
double stabIn  = 600.;                  /* Stabilization time for In               */
double stabSb  = 600.;                  /* Stabilization time for Sb               */

double TGamax   = 999.;                /* Upper Ga  temp limit                     */
double TAl3max  = 1249.;                /* Upper Al3 temp limit                     */
double TAl4max  = 1249.;                /* Upper Al4 temp limit                     */
double TInmax   =  899.;                /* Upper In  temp limit                     */
double TSbmax   =  799.;                /* Upper Sb  temp limit                     */
double TSimax   = 1399.;                /* Upper Si  temp limit                     */
double TBemax   = 1149.;                /* Upper Be  temp limit                     */

double TGamin   =  600.;                /* Lower Ga  temp limit                     */
double TAl3min  =  820.;                /* Lower Al3 temp limit                     */
double TAl4min  =  820.;                /* Lower Al4 temp limit                     */
double TInmin   =  400.;                /* Lower In  temp limit                     */
double TSbmin   =  200.;                /* Lower Sb  temp limit                     */
double TSimin   =  400.;                /* Lower Si  temp limit                     */
double TBemin   =  400.;                /* Lower Be  temp limit                     */

double TGa_outg;                        /* Outgas temp for Ga                       */
double TAl3_outg;                       /* Outgas temp for Al3                      */
double TAl4_outg;                       /* Outgas temp for Al4                      */
double TIn_outg;                        /* Outgas temp for In                       */
double TSi_outg;                        /* Outgas temp for Si                       */
double TBe_outg;                        /* Outgas temp for Be                       */
double TSb_outg;                        /* Outgas temp for Sb                       */

double fluxGa;                          /* Measured flux Ga                         */
double fluxAl3;                         /* Measured flux Al3                        */
double fluxAl4;                         /* Measured flux Al4                        */
double fluxIn;                          /* Measured flux In                         */
double fluxSb;                          /* Measured flux Sb                         */

double TGa;                             /* Temp for Ga                              */
double TAl3;                            /* Temp for Al3                             */
double TAl4;                            /* Temp for Al4                             */
double TIn;                             /* Temp for In                              */
double TSb;                             /* Temp for Sb                              */

double errorGa;                         /* Ga  flux error                           */
double errorAl3;                        /* Al3 flux error                           */
double errorAl4;                        /* Al4 flux error                           */
double errorIn;                         /* In flux error                            */
double errorSb;                         /* Sb flux error                            */

int doneGa;                             /* Completion flag for Ga  (1=yes 0 = no)   */
int doneAl3;                            /* Completion flag for Al3 (1=yes 0 = no)   */
int doneAl4;                            /* Completion flag for Al4 (1=yes 0 = no)   */
int doneIn;                             /* Completion flag for In  (1=yes 0 = no)   */
```

```
int doneSb;                              /* Completion flag for Sb  (1=yes 0 = no)    */

int Tbeamread = 1;                       /* Time between flux readings (sec)          */
int iread;                               /* counter during flux sampling              */
int attempt;                             /* Number of changes of cell temp            */
int log_id1;                              /* Data logger                              */

double beam_flux;                        /* Measured flux value                       */
double sum_flux;                         /* Summation variable for flux calc          */
double beam_flux_open;                   /* Beam flux with open shutter               */
double beam_flux_close;                  /* Beam flux with close shutter              */
double background;                       /* Background pressure after shutter closed */
long tnow;                               /* Present time                              */
long tbegin;                             /* Time to start up-ramp                     */
long tleft;                              /* Time left before start of up-ramp         */
/**/
/*=====================================================================*/
/*========== Wait until start-up time        ==========================*/
/*=====================================================================*/
/**/
echo();
echo(" ===================================================== ");
echo(" ||  DON'T FORGET TO PUT SHUTTERS AND CAR IN REMOTE  || ");
echo(" ===================================================== ");
echo();

targ_time = hour * 60 + minute;                 /* Timer code from J. Vlcek 5/5/97 */
fd = open( "_clock", O_RDONLY );

if ( fd < 0 ) {
  echo( "I'm sorry, but I'm unable to access the system clock for some reason." );
  echo( "I am unable to schedule your command file for later execution." );
  echo( "Please send an email to software@epimbe.com describing this problem." );
  exit( EXIT_FAILURE );
}
echo();
echo( " The up-ramp is now scheduled for execution." );

start_time = ioctl( fd, CLIOCTOD, 0 );

/* Wait until midnight if the target time is earlier in the day
 * than the current time (ie, the file executes tomorrow).
 */
if ( start_time >= targ_time ) {
  while ( ioctl( fd, CLIOCTOD, 0 ) >= start_time )
  {
    sleep( 20.0 );
  }
 }

while ( ioctl( fd, CLIOCTOD, 0 ) < targ_time )
  {
    sleep( 20.0 );
  }
close( fd );

/**/
/*=================================================================*/
/*========== Test chamber pressure before cell up ramps ==============*/
/*=================================================================*/
/**/
echo(" Testing chamber pressure");
load ("pos3.cmd");                                   /*  Turn CAR */
echo(" Wait 25 sec for substrate to turn toward cells");
echo();
sleep ( 25 );
iread = 0;
sum_flux = 0;
while (iread<10)
     {
     beam_flux = reading(flux);
     echo(" Flux gauge = ", beam_flux);
     if(beam_flux > 0)
            {
            sum_flux = sum_flux + beam_flux;
            iread = iread+1;
            }
```

```
     sleep( Tbeamread );
     }
 background = sum_flux/10;

 if( background < backtest && background > 1.0E-11 )
          {
          echo();
          echo(" The chamber pressure    ", background );
          echo(" passed the test limit   ", backtest   );
          echo(" The cells will now be ramped up ");
          echo();
          }
          else
          {
          echo();
          echo(" The chamber pressure    ", background );
          echo(" exceeds the test limit  ", backtest   );
          echo(" I will not ramp up the cells ");
          echo();
          kill( getpid(), SIGKILL);
          }
/**/
/*================================================================*/
/*=========== Ramp up cells with dummy block facing cells ============*/
/*================================================================*/
/**/
/**/
/*———— Start data logger   ————*/
/**/
log_id1 = logger(20.0,                          /* log every 20 seconds */
            't',
            'temp(subs)',
            'temp(Ga)',
            'temp(Al3)',
            'temp(Al4)',
            'temp(In)',
            'temp(Si)',
            'temp(Be)',
            'is_open(Ga)',
            'is_open(Al3)',
            'is_open(Al4)',
            'is_open(In)',
            'is_open(Si)',
            'is_open(Be)',
            'reading(flux)',
            "fluxcal.dat");

/*  PUT SB BACK WHEN THE EUROTHERM IS BACK, */
/**/
/*———— Turn CAR and check if ramping should be done  ————*/
/**/
load ("pos3.cmd");                                          /*  Turn CAR */
echo(" Wait 25 sec for substrate to turn toward cells");
echo();
sleep ( 25 );

if ( rampGa  == 1  || rampAl3 == 1 || rampAl4 == 1 || rampIn  == 1 ||     /* If any cell is to be ramped */
     rampSi  == 1  || rampBe  == 1 || rampSb  == 1 )
     {
/**/
/*—— Ramp up to outgas temperature  (2.5% above nominal temp)  ——*/
/**/
      echo(" Wait ",ramptime, " min for cells to ramp up ");
      echo();
      if ( rampGa  == 1 ) TGa  = temp( Ga  );                    /*  Get current setpoints */
      if ( rampAl3 == 1 ) TAl3 = temp( Al3 );
      if ( rampAl4 == 1 ) TAl4 = temp( Al4 );
      if ( rampIn  == 1 ) TIn  = temp( In  );
      if ( rampSi  == 1 ) TSi  = temp( Si  );
      if ( rampBe  == 1 ) TBe  = temp( Be  );
      if ( rampSb  == 1 ) TSb  = temp( Sb  );

      TGa_outg  = setpGa *(1.0 + outgas/100.0);                  /* Create outgas temp  */
      TAl3_outg = setpAl3*(1.0 + outgas/100.0);
      TAl4_outg = setpAl4*(1.0 + outgas/100.0);
      TIn_outg  = setpIn *(1.0 + outgas/100.0);
      TSi_outg  = setpSi *(1.0 + outgas/100.0);
```

```
    TBe_outg  = setpBe *(1.0 + outgas/100.0);
    TSb_outg  = setpSb *(1.0 + outgas/100.0);

    if ( TGa_outg  > TGamax  )
        {
        TGa_outg  = TGamax;
        echo(" Warning - Ga will be outgassed at max temp ", TGamax, " C");
        }
    if ( TAl3_outg > TAl3max )
        {
        TAl3_outg = TAl3max;
        echo(" Warning - Al3 will be outgassed at max temp ", TAl3max," C");
        }
    if ( TAl4_outg > TAl4max )
        {
        TAl4_outg = TAl4max;
        echo(" Warning - Al4 will be outgassed at max temp ", TAl4max," C");
        }
    if ( TIn_outg  > TInmax  )
        {
        TIn_outg  = TInmax;
        echo(" Warning - In will be outgassed at max temp ", TInmax," C");
        }
    if ( TSi_outg  > TSimax  )
        {
        TSi_outg  = TSimax;
        echo(" Warning - Si will be outgassed at max temp ", TSimax," C");
        }
    if ( TBe_outg  > TBemax  )
        {TBe_outg  = TBemax;
        echo(" Warning - Be will be outgassed at max temp ", TBemax," C");
        }
    if ( TSb_outg  > TSbmax  )
        {
        TSb_outg  = TSbmax;
        echo(" Warning - Sb will be outgassed at max temp ", TSbmax," C");
        }

    if ( rampGa  == 1 ) set_ramp( Ga ,(TGa_outg  - TGa )/ramptime );    /* Set new ramp rates */
    if ( rampAl3 == 1 ) set_ramp( Al3,(TAl3_outg - TAl3)/ramptime );
    if ( rampAl4 == 1 ) set_ramp( Al4,(TAl4_outg - TAl4)/ramptime );
    if ( rampIn  == 1 ) set_ramp( In ,(TIn_outg  - TIn )/ramptime );    /* DEG/MIN            */
    if ( rampSi  == 1 ) set_ramp( Si ,(TSi_outg  - TSi )/ramptime );
    if ( rampBe  == 1 ) set_ramp( Be ,(TBe_outg  - TBe )/ramptime );
    if ( rampSb  == 1 ) set_ramp( Sb ,(TSb_outg  - TSb )/ramptime );

    if ( rampGa  == 1 ) set_temp( Ga ,TGa_outg);                       /* Set new temperatures */
    if ( rampAl3 == 1 ) set_temp( Al3,TAl3_outg);
    if ( rampAl4 == 1 ) set_temp( Al4,TAl4_outg);
    if ( rampIn  == 1 ) set_temp( In ,TIn_outg);
    if ( rampSi  == 1 ) set_temp( Si ,TSi_outg);
    if ( rampBe  == 1 ) set_temp( Be ,TBe_outg);
    if ( rampSb  == 1 ) set_temp( Sb ,TSb_outg);

    sleep ( ramptime*60);                                  /* Wait until ramp completed */
/**/
/*—— Open shutters and outgas   ——*/
/**/
    echo(" Outgas cells ",outgastime, " min");
    echo();
    if ( rampGa  == 1 ) shopen(Ga );
    if ( rampAl3 == 1 ) shopen(Al3);
    if ( rampAl4 == 1 ) shopen(Al4);
    if ( rampIn  == 1 ) shopen(In );
    if ( rampSi  == 1 ) shopen(Si );
    if ( rampBe  == 1 ) shopen(Be );
    if ( rampSb  == 1 ) shopen(Sb );

    sleep ( outgastime*60);
/**/
/*—— Close shutters and ramp down to setpoints   ——*/
/**/
    echo(" Close cells and ramp to setpoints wait 5 min");
    echo();
    if ( rampGa  == 1 ) shclose(Ga );                                  /* Close the shutters */
    if ( rampAl3 == 1 ) shclose(Al3);
    if ( rampAl4 == 1 ) shclose(Al4);
```

```
      if ( rampIn  == 1 ) shclose(In );
      if ( rampSi  == 1 ) shclose(Si );
      if ( rampBe  == 1 ) shclose(Be );
      if ( rampSb  == 1 ) shclose(Sb );

      if ( rampGa  == 1 ) TGa  = temp( Ga  );                          /*  Get current setpoints */
      if ( rampAl3 == 1 ) TAl3 = temp( Al3 );
      if ( rampAl4 == 1 ) TAl4 = temp( Al4 );
      if ( rampIn  == 1 ) TIn  = temp( In  );
      if ( rampSi  == 1 ) TSi  = temp( Si  );
      if ( rampBe  == 1 ) TBe  = temp( Be  );
      if ( rampSb  == 1 ) TSb  = temp( Sb  );

      if ( rampGa  == 1 ) set_ramp( Ga ,(TGa  - setpGa )/5. );      /* Set new ramp rates */
      if ( rampAl3 == 1 ) set_ramp( Al3,(TAl3 - setpAl3)/5. );
      if ( rampAl4 == 1 ) set_ramp( Al4,(TAl4 - setpAl4)/5. );
      if ( rampIn  == 1 ) set_ramp( In ,(TIn  - setpIn )/5. );      /* DEG/MIN          */
      if ( rampSi  == 1 ) set_ramp( Si ,(TSi  - setpSi )/5. );
      if ( rampBe  == 1 ) set_ramp( Be ,(TBe  - setpBe )/5. );
      if ( rampSb  == 1 ) set_ramp( Sb ,(TSb  - setpSb )/5. );

      if ( rampGa  == 1 ) set_temp( Ga ,setpGa  );                     /* Set new temperatures */
      if ( rampAl3 == 1 ) set_temp( Al3,setpAl3 );
      if ( rampAl4 == 1 ) set_temp( Al4,setpAl4 );
      if ( rampIn  == 1 ) set_temp( In ,setpIn  );
      if ( rampSi  == 1 ) set_temp( Si ,setpSi  );
      if ( rampBe  == 1 ) set_temp( Be ,setpBe  );
      if ( rampSb  == 1 ) set_temp( Sb ,setpSb  );

      sleep ( 300 );
/**/
/*---- Wait 5 more minutes to ensure stability      ----*/
/**/
      echo(" Wait 5 min for stability ");
      echo();
      sleep ( 300 );
      }                                                         /* End of cell excercise */
/**/
/*=================================================================*/
/*================       Measure Ga flux     ========================*/
/*=================================================================*/
/**/
if( calGa == 1 )
   {
    doneGa = 0;
    attempt = 0;
    echo(" Measure Ga flux");
    echo();
    TGa = setpGa;
    set_ramp(Ga,100);                                           /* set fast rate for small adjustments */
    }
while ( doneGa == 0 && calGa == 1 )
{
/**/
/*---- Set new temperature turn flux guage away from cells and wait for stabilization     ----*/
/**/
   if( TGa < TGamax && TGa > TGamin )
         set_temp(Ga,TGa);
         else
         {
         echo(" Ga setpoint outside allowd interval - Process terminated");
         kill( getpid(), SIGKILL);
         }
   attempt = attempt + 1;
   echo(" Seting new temp = ",TGa," and waiting ",stabGa," sec.");
   echo(" Time is: ",mctime(time(0)) );
   echo();
   load ("pos3.cmd");
   sleep ( stabGa );
/**/
/*---- , open shutter and wait ----*/
/**/
   shopen(Ga);
   echo(" Wait ",transient," sec during transient");
   echo();
   sleep ( transient );
   load ("pos1.cmd");
```

```
   echo(" Wait 30 sec for guage to turn toward cells");
   echo();
   sleep ( 30 );

/**/
/*---------- Measure with shutter open -----------*/
/**/
   echo(" Measure with shutter open");
   echo();
   iread = 0;
   sum_flux = 0;
   while (iread<10)
       {
       beam_flux = reading(flux);
       echo(" Flux gauge = ", beam_flux);
       if(beam_flux > 0)
              {
              sum_flux = sum_flux + beam_flux;
              iread = iread+1;
              }
       sleep( Tbeamread );
       }
   beam_flux_open = sum_flux/10;
   echo();
   echo(" Average Flux = ", beam_flux_open);
   echo();
/**/
/*---------- Measure with shutter closed -----------*/
/**/
   shclose(Ga);
   sleep(20);
   iread = 0;
   sum_flux = 0;
   while (iread<10)
       {
       back_flux = reading(flux);
       echo(" Flux gauge = ", back_flux);
       if(back_flux > 0 )
              {
              sum_flux = sum_flux + back_flux;
              iread = iread+1;
              }
       sleep( Tbeamread );
       }
   beam_flux_close = sum_flux/10;
   echo();
   echo(" Average Flux = ", beam_flux_close);
   echo();

   fluxGa = beam_flux_open - beam_flux_close;
   echo(" Net Flux = ", fluxGa);
   echo();
/**/
/*---------- Test flux and calc temp correction----------*/
/**/
   errorGa = ( fluxGa-targetGa )/targetGa;
   echo(" Ga error = ",errorGa );
   if( fabs( errorGa ) > precGa )
          {
          TGa = 1./(1./(TGa+273) - (log10(fluxGa)-log10(targetGa))/slopeGa ) - 273;
          }
          else
          doneGa = 1;
}
/**/
/*---------- Ga calibrated ----------*/
/**/
if( calGa == 1 )
    {
    set_ramp(Ga,10);                                    /* reset slow rate for protection */
    echo();
    echo(" Ga calibration converged in ",attempt," attempts");
    echo(" Final error was ",errorGa );
    echo();
    }
/**/
/*==================================================================*/
```

```
/*================      End of Ga loop      ========================*/
/*================================================================*/
/**/
/**/
/*================================================================*/
/*================      Measure Al3 flux      ========================*/
/*================================================================*/
/**/
if( calAl3 == 1 )
    {
    doneAl3 = 0;
    attempt = 0;
    echo(" Measure Al3 flux");
    echo();
    TAl3 = setpAl3;
    set_ramp(Al3,100);                              /* set fast rate for small adjustments */
    }
while ( doneAl3 == 0 && calAl3 == 1 )
{
/**/
/*—— Set new temperature, turn flux guage away from cells and wait for stabilization     ——*/
/**/
   if( TAl3 < TAl3max && TAl3 > TAl3min )
        set_temp(Al3,TAl3);
        else
        {
        echo(" Al3 setpoint outside allowd interval - Process terminated");
        kill( getpid(), SIGKILL);
        }
   attempt = attempt + 1;
   echo(" Seting new temp = ",TAl3," and waiting ",stabAl3," sec");
   echo(" Time is: ",mctime(time(0)) );
   echo();
   load ("pos3.cmd");
   sleep ( stabAl3 );
/**/
/*—— , open shutter and wait ——*/
/**/
   shopen(Al3);
   echo(" Wait ",transient," sec during transient");
   echo();
   sleep ( transient );
   load ("pos1.cmd");
   echo(" Wait 30 sec for guage to turn toward cells");
   echo();
   sleep ( 30 );

/**/
/*———— Measure with shutter open ————*/
/**/
   echo(" Measure with shutter open");
   echo();
   iread = 0;
   sum_flux = 0;
   while (iread<10)
       {
       beam_flux = reading(flux);
       echo(" Flux gauge = ", beam_flux);
       if(beam_flux > 0)
             {
             sum_flux = sum_flux + beam_flux;
             iread = iread+1;
             }
       sleep( Tbeamread );
       }
   beam_flux_open = sum_flux/10;
   echo();
   echo(" Average Flux = ", beam_flux_open);
   echo();
/**/
/*———— Measure with shutter closed ————*/
/**/
   shclose(Al3);
   sleep(20);
   iread = 0;
   sum_flux = 0;
   while (iread<10)
```

```
        {
        back_flux = reading(flux);
        echo(" Flux gauge = ", back_flux);
        if(back_flux > 0 )
                {
                sum_flux = sum_flux + back_flux;
                iread = iread+1;
                }
        sleep( Tbeamread );
        }
    beam_flux_close = sum_flux/10;
    echo();
    echo(" Average Flux = ", beam_flux_close);
    echo();

    fluxAl3 = beam_flux_open - beam_flux_close;
    echo(" Net Flux = ", fluxAl3);
    echo();
/**/
/*---------- Test flux and calc temp correction-----------*/
/**/
    errorAl3 = ( fluxAl3-targetAl3 )/targetAl3;
    echo(" Al3 error = ",errorAl3 );
    if( fabs( errorAl3 ) > precAl3 )
            {
            TAl3 = 1./(1./(TAl3+273) - (log10(fluxAl3)-log10(targetAl3))/slopeAl3 ) - 273;
             }
             else
             doneAl3 = 1;
}
/**/
/*---------- Al3 calibrated ----------*/
/**/
if( calAl3 == 1 )
     {
      set_ramp(Al3,10);                                      /* reset slow rate for protection */
      echo();
      echo(" Al3 calibration converged in ",attempt," attempts");
      echo(" Final error was ",errorAl3 );
      echo();
      }
/**/
/*=================================================================*/
/*================       End of Al3 loop       =========================*/
/*=================================================================*/
/**/
/*=================================================================*/
/*================       Measure Al4 flux      =========================*/
/*=================================================================*/
/**/
if( calAl4 == 1 )
     {
      doneAl4 = 0;
      attempt = 0;
      echo(" Measure Al4 flux");
      echo();
      TAl4 = setpAl4;
      set_ramp(Al4,100);                                     /* set fast rate for small adjustments */
      }
while ( doneAl4 == 0 && calAl4 == 1 )
{
/**/
/*----- Set new temperature, turn flux guage away from cells and wait for stabilization      -----*/
/**/
    if( TAl4 < TAl4max && TAl4 > TAl4min )
          set_temp(Al4,TAl4);
          else
          {
          echo(" Al4 setpoint outside allowd interval - Process terminated");
          kill( getpid(), SIGKILL);
          }
    attempt = attempt + 1;
    echo(" Seting new temp = ",TAl4," and waiting ",stabAl4," sec");
    echo(" Time is: ",mctime(time(0)) );
    echo();
    load ("pos3.cmd");
    sleep ( stabAl4 );
```

```
/**/
/*----  open shutter and wait ----*/
/**/
   shopen(Al4);
   echo(" Wait ",transient," sec during transient");
   echo();
   sleep ( transient );
   load ("pos1.cmd");
   echo(" Wait 30 sec for guage to turn toward cells");
   echo();
   sleep ( 30 );

/**/
/*---------- Measure with shutter open ------------*/
/**/
   echo(" Measure with shutter open");
   echo();
   iread = 0;
   sum_flux = 0;
   while (iread<10)
      {
      beam_flux = reading(flux);
      echo(" Flux gauge = ", beam_flux);
      if(beam_flux > 0)
            {
            sum_flux = sum_flux + beam_flux;
            iread = iread+1;
            }
      sleep( Tbeamread );
      }
   beam_flux_open = sum_flux/10;
   echo();
   echo("Average Flux = ", beam_flux_open);
   echo();
/**/
/*---------- Measure with shutter closed ------------*/
/**/
   shclose(Al4);
   sleep(20);
   iread = 0;
   sum_flux = 0;
   while (iread<10)
      {
      back_flux = reading(flux);
      echo(" Flux gauge = ", back_flux);
      if(back_flux > 0 )
            {
            sum_flux = sum_flux + back_flux;
            iread = iread+1;
            }
      sleep( Tbeamread );
      }
   beam_flux_close = sum_flux/10;
   echo();
   echo(" Average Flux = ", beam_flux_close);
   echo();

   fluxAl4 = beam_flux_open - beam_flux_close;
   echo(" Net Flux = ", fluxAl4);
   echo();
/**/
/*---------- Test flux and calc temp correction-----------*/
/**/
   errorAl4 = ( fluxAl4-targetAl4 )/targetAl4;
   echo(" Al4 error = ",errorAl4 );
   if( fabs( errorAl4 ) > precAl4 )
          {
          TAl4 = 1./(1./(TAl4+273) - (log10(fluxAl4)-log10(targetAl4))/slopeAl4 ) - 273;
           }
           else
           doneAl4 = 1;
}
/**/
/*---------- Al4 calibrated ----------*/
/**/
if( calAl4 == 1 )
    {
```

```
     set_ramp(Al4,10);                                              /* reset slow rate for protection */
     echo();
     echo(" Al4 calibration converged in ",attempt," attempts");
     echo(" Final error was ",errorAl4 );
     echo();
     }
/**/
/*===================================================================*/
/*================         End of Al4 loop       ========================*/
/*===================================================================*/
/**/
/*===================================================================*/
/*================         Measure In flux      ========================*/
/*===================================================================*/
/**/
if( calIn == 1 )
    {
    doneIn = 0;
    attempt = 0;
    echo(" Measure In flux");
    echo();
    TIn = setpIn;
    set_ramp(In,100);                                               /* set fast rate for small adjustments */
    }
while ( doneIn == 0 && calIn == 1 )
{
/**/
/*—— Set new temperature, turn flux guage away from cells and wait for stabilization     ——*/
/**/
   if( TIn < TInmax && TIn > TInmin )
         set_temp(In,TIn);
         else
         {
         echo(" In setpoint outside allowd interval - Process terminated");
         kill( getpid(), SIGKILL);
         }
   attempt = attempt + 1;
   echo(" Seting new temp = ",TIn," and waiting ",stabIn," sec");
   echo(" Time is: ",mctime(time(0)) );
   echo();
   load ("pos3.cmd");
   sleep ( stabIn );
/**/
/*——   open shutter and wait ——*/
/**/
   shopen(In);
   echo(" Wait ",transient," sec during transient");
   echo();
   sleep ( transient );
   load ("pos1.cmd");
   echo(" Wait 30 sec for guage to turn toward cells");
   echo();
   sleep ( 30 );
/**/
/*———— Measure with shutter open ————*/
/**/
   echo(" Measure with shutter open");
   echo();
   iread = 0;
   sum_flux = 0;
   while (iread<10)
        {
        beam_flux = reading(flux);
        echo(" Flux gauge = ", beam_flux);
        if(beam_flux > 0)
                {
                sum_flux = sum_flux + beam_flux;
                iread = iread+1;
                }
        sleep( Tbeamread );
        }
   beam_flux_open = sum_flux/10;
   echo();
   echo(" Average Flux = ", beam_flux_open);
   echo();
/**/
/*———— Measure with shutter closed ————*/
```

```
/**/
   shclose(In);
   sleep(20);
   iread = 0;
   sum_flux = 0;
   while (iread<10)
        {
        back_flux = reading(flux);
        echo(" Flux gauge = ", back_flux);
        if(back_flux > 0 )
               {
               sum_flux = sum_flux + back_flux;
               iread = iread+1;
               }
        sleep( Tbeamread );
        }
   beam_flux_close = sum_flux/10;
   echo();
   echo(" Average Flux = ", beam_flux_close);
   echo();

   fluxIn = beam_flux_open - beam_flux_close;
   echo(" Net Flux = ", fluxIn);
   echo();
/**/
/*———————— Test flux and calc temp correction————————*/
/**/
   errorIn = ( fluxIn-targetIn )/targetIn;
   echo(" In error = ",errorIn );
   if( fabs( errorIn ) > precIn )
           {
           TIn = 1./(1./(TIn+273) - (log10(fluxIn)-log10(targetIn))/slopeIn ) - 273;
            }
            else
            doneIn = 1;
}
/**/
/*———————— In calibrated ————————*/
/**/
if( calIn == 1 )
    {
     set_ramp(In,10);                                                /* reset slow rate for protection */
     echo();
     echo("In calibration converged in ",attempt," attempts");
     echo("Final error was ",errorIn );
     echo();
     }
/**/
/*=================================================================*/
/*================      End of In loop      ========================*/
/*=================================================================*/
/**/
/*=================================================================*/
/*================      Measure Sb flux     ========================*/
/*=================================================================*/
/**/
if( calSb == 1 )
    {
     doneSb = 0;
     attempt = 0;
     echo(" Measure Sb flux");
     echo();
     TSb = setpSb;
     set_ramp(Sb,100);                                               /* set fast rate for small adjustments */
     }
while ( doneSb == 0 && calSb == 1 )
{
/**/
/*——— Set new temperature, turn flux guage away from cells and wait for stabilization     ———*/
/**/
   if( TSb < TSbmax && TSb > TSbmin )
         set_temp(Sb,TSb);
         else
         {
         echo(" Sb setpoint outside allowd interval - Process terminated");
         kill( getpid(), SIGKILL);
         }
   attempt = attempt + 1;
```

```
   echo(" Seting new temp = ",TSb," and waiting ",stabSb," sec");
   echo(" Time is: ",mctime(time(0)) );
   echo();
   load ("pos3.cmd");
   sleep ( stabSb );
/**/
/*----  open shutter and wait ----*/
/**/
   shopen(Sb);
   echo(" Wait ",transient," sec during transient");
   echo();
   sleep ( transient );
   load ("pos1.cmd");
   echo(" Wait 30 sec for guage to turn toward cells");
   echo();
   sleep ( 30 );

/**/
/*---------- Measure with shutter open ------------*/
/**/
   echo(" Measure with shutter open");
   echo();
   iread = 0;
   sum_flux = 0;
   while (iread<10)
       {
       beam_flux = reading(flux);
       echo(" Flux gauge = ", beam_flux);
       if(beam_flux > 0)
              {
              sum_flux = sum_flux + beam_flux;
              iread = iread+1;
              }
       sleep( Tbeamread );
       }
   beam_flux_open = sum_flux/10;
   echo();
   echo(" Average Flux = ", beam_flux_open);
   echo();
/**/
/*---------- Measure with shutter closed ------------*/
/**/
   shclose(Sb);
   sleep(20);
   iread = 0;
   sum_flux = 0;
   while (iread<10)
       {
       back_flux = reading(flux);
       echo(" Flux gauge = ", back_flux);
       if(back_flux > 0 )
              {
              sum_flux = sum_flux + back_flux;
              iread = iread+1;
              }
       sleep( Tbeamread );
       }
   beam_flux_close = sum_flux/10;
   echo();
   echo(" Average Flux = ", beam_flux_close);
   echo();

   fluxSb = beam_flux_open - beam_flux_close;
   echo(" Net Flux = ", fluxSb);
   echo();
/**/
/*---------- Test flux and calc temp correction----------*/
/**/
   errorSb = ( fluxSb-targetSb )/targetSb;
   echo(" Sb error = ",errorSb );
   if( fabs( errorSb ) > precSb )
           {
           TSb = 1./(1./(TSb+273) - (log10(fluxSb)-log10(targetSb))/slopeSb ) - 273;
            }
            else
            doneSb = 1;
}
```

```
/**/
/*———— Sb calibrated ————*/
/**/
if( calSb == 1 )
    {
    set_ramp(Sb,10);                                      /* reset slow rate for protection */
    echo();
    echo(" Sb calibration converged in ",attempt," attempts");
    echo(" Final error was ",errorSb );
    echo();
    }
/**/
/*=================================================================*/
/*================      End of Sb loop      ========================*/
/*=================================================================*/
/**/
/**/
/*=================================================================*/
/*================      Summary             ========================*/
/*=================================================================*/
/**/
kill(log_id1, SIGTERM);                    /* Stop logging */
echo();
if( calGa  == 1 ) echo(" Final Ga  error was ",errorGa , " TGa  = ",TGa );
if( calAl3 == 1 ) echo(" Final Al3 error was ",errorAl3, " TAl3 = ",TAl3);
if( calAl4 == 1 ) echo(" Final Al4 error was ",errorAl4, " TAl4 = ",TAl4);
if( calIn  == 1 ) echo(" Final In  error was ",errorIn , " TIn  = ",TIn );
if( calSb  == 1 ) echo(" Final Sb  error was ",errorSb , " TSb  = ",TSb );
echo();
echo(" A record of the temperatures, shutter status and flux values ");
echo(" is stored in the file FLUXCAL.DAT ");
echo();
```

Distribution

Admnstr
Defns Techl Info Ctr
ATTN DTIC-OCP
8725 John J Kingman Rd Ste 0944
FT Belvoir VA 22060-6218

DARPA
ATTN S Welby
3701 N Fairfax Dr
Arlington VA 22203-1714

Ofc of the Secy of Defns
ATTN ODDRE (R&AT)
The Pentagon
Washington DC 20301-3080

Ofc of the Secy of Defns
ATTN OUSD(A&T)/ODDR&E(R) R J Trew
3080 Defense Pentagon
Washington DC 20301-7100

AMCOM MRDEC
ATTN AMSMI-RD W C McCorkle
Redstone Arsenal AL 35898-5240

US Army TRADOC
Battle Lab Integration & Techl Dirctrt
ATTN ATCD-B
FT Monroe VA 23651-5850

US Military Acdmy
Dept of Mathematical Sci
ATTN MAJ L G Eggen
West Point NY 10996-1786

US Military Acdmy
Mathematical Sci Ctr of Excellence
ATTN MADN-MATH MAJ M Huber
Thayer Hall
West Point NY 10996-1786

Dir for MANPRINT
Ofc of the Deputy Chief of Staff for Prsnnl
ATTN J Hiller
The Pentagon Rm 2C733
Washington DC 20301-0300

SMC/CZA
2435 Vela Way Ste 1613
El Segundo CA 90245-5500

US Army ARDEC
ATTN AMSTA-AR-TD
Bldg 1
Picatinny Arsenal NJ 07806-5000

US Army Info Sys Engrg Cmnd
ATTN AMSEL-IE-TD F Jenia
FT Huachuca AZ 85613-5300

US Army Natick RDEC Acting Techl Dir
ATTN SBCN-T P Brandler
Natick MA 01760-5002

US Army Simulation Train & Instrmntn Cmnd
ATTN AMSTI-CG M Macedonia
ATTN J Stahl
12350 Research Parkway
Orlando FL 32826-3726

US Army Tank-Automtv Cmnd RDEC
ATTN AMSTA-TR J Chapin
Warren MI 48397-5000

Nav Surfc Warfare Ctr
ATTN Code B07 J Pennella
17320 Dahlgren Rd Bldg 1470 Rm 1101
Dahlgren VA 22448-5100

Hicks & Assoc Inc
ATTN G Singley III
1710 Goodrich Dr Ste 1300
McLean VA 22102

Palisades Inst for Rsrch Svc Inc
ATTN E Carr
1745 Jefferson Davis Hwy Ste 500
Arlington VA 22202-3402

Director
US Army Rsrch Lab
ATTN AMSRL-RO-D JCI Chang
ATTN AMSRL-RO-EN W D Bach
PO Box 12211
Research Triangle Park NC 27709

US Army Rsrch Lab
ATTN AMSRL-CI-AI-R Mail & Records Mgmt
ATTN AMSRL-CI-AP Techl Pub (2 copies)

Distribution (cont'd)

US Army Rsrch Lab (cont'd)
ATTN AMSRL-CI-LL Techl Lib (2 copies)
ATTN AMSRL-D D R Smith
ATTN AMSRL-DD J M Miller

US Army Rsrch Lab (cont'd)
ATTN AMSRL-SE-RL S Svensson (4 copies)
Adelphi MD 20783-1197

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE February 2001	3. REPORT TYPE AND DATES COVERED Final, January-June 2000

4. TITLE AND SUBTITLE Process for Automated, Safe MBE Start and Flux Calibration

5. FUNDING NUMBERS
DA PR: AH94
PE: 62705A

6. AUTHOR(S) Stefan Svensson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
Attn: AMSRL-SE-RL email: svensson@arl.army.mil
2800 Powder Mill Road
Adelphi, MD 20783-1197

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-2361

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
2800 Powder Mill Road
Adelphi, MD 20783-1197

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
ARL PR: ONE6J2
AMS code: 622705.H94

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A command procedure has been developed for the U.S. Army Research Laboratory (ARL) molecular beam epitaxy (MBE) computer control system that allows a user to set up the system for an automated, unattended start each morning. The automated sequence consists of-

1. A system safety check to determine if cell ramping should be allowed.
2. A cell temperature ramp to an outgassing temperature.
3. An outgassing of cells.
4. A ramp-down of cells to nominal operating temperatures.
5. An automated setup through an iterative process of flux measurements and changes of temperatures until desired targets are reached.

This command procedure allows a daily, safe start-up of the MBE system and generates identical flux settings that improve the crystal growth reproducibility. Typically, one can save two hours or more of a work day by using this automated procedure.

14. SUBJECT TERMS Computer control, molecular beam epitaxy	15. NUMBER OF PAGES 31
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102